ARMY RESEARCH LABORATORY

Natural Computing: Analysis of Graphs for Computer Representation

Som Karamchetty

ARL-TR-2042

February 2000

Abstract

Some fundamental objects in practical documents have not been implemented in software so that they can be used easily for calculation. One such object is the graph. A survey of practical graphs found in a variety of real-world documents reveals that many of their useful features are not captured in software. I describe the salient characteristics and features of graphs and propose data structures and computer representations for graph objects. Through the adoption of such structures and representations, practical graph objects could be developed for use by domain specialists. Such graphs embedded in electronic documents can be used in interactive applications to retrieve data, but most importantly, they could be used as functional representations for "cutting and pasting" in procedures and programs. Use of these graph objects, together with other natural computing objects (such as equations, tables, and procedures), will permit electronic documents like handbooks, textbooks, journals, and bulletins to be used seamlessly for calculations by both domain specialists and naive users. Such developments will reduce the lag between information availability and its use in calculations, encouraging the further development of knowledge. The face of software development for computation will change, and many of the software engineering costs will be contained.

Contents

Figures

1. Introduction

In another report (Karamchetty, 2000a), I described Natural Computing (Karamchetty, 1997) as an approach to software for computation that is based on the way people actually compute, using available information in a variety of formats. In a separate report (Karamchetty, 2000b), I described the use of tables in Natural Computing. This report builds on those previous discussions by examining one aspect of Natural Computing: the use of graphs.

Before the invention of computers, most knowledge was captured in the form of books and other paper documents. (Books can be further classified as textbooks, reference books, handbooks, and journals, based on the temporal nature of the information. More transient documents are flyers, brochures, and receipts.) Information was printed on paper for storage, retrieval, and communication. The paper-based information was read by the end user. By reading the information from one or more documents and by combining it with one's own intuition, invention, and discovery, one generated new information and wrote (printed) it in the form of another paper document.

In the paper-based universe, when we dealt with technical matter, domain knowledge was captured in the form of text containing equations, tables, graphs, and pictures. Without pictures, descriptions of scenes were elaborate. As we recall, "a picture is worth a thousand words." In technical subjects, the pictures could be sketches, schematics, drawings, paintings, or photographs. Sketches stood for descriptions of parts and components in terms of shapes and sizes. Schematics and other diagrams showed the mutual relationships of components in a system and the state of the system and its temporal variations.

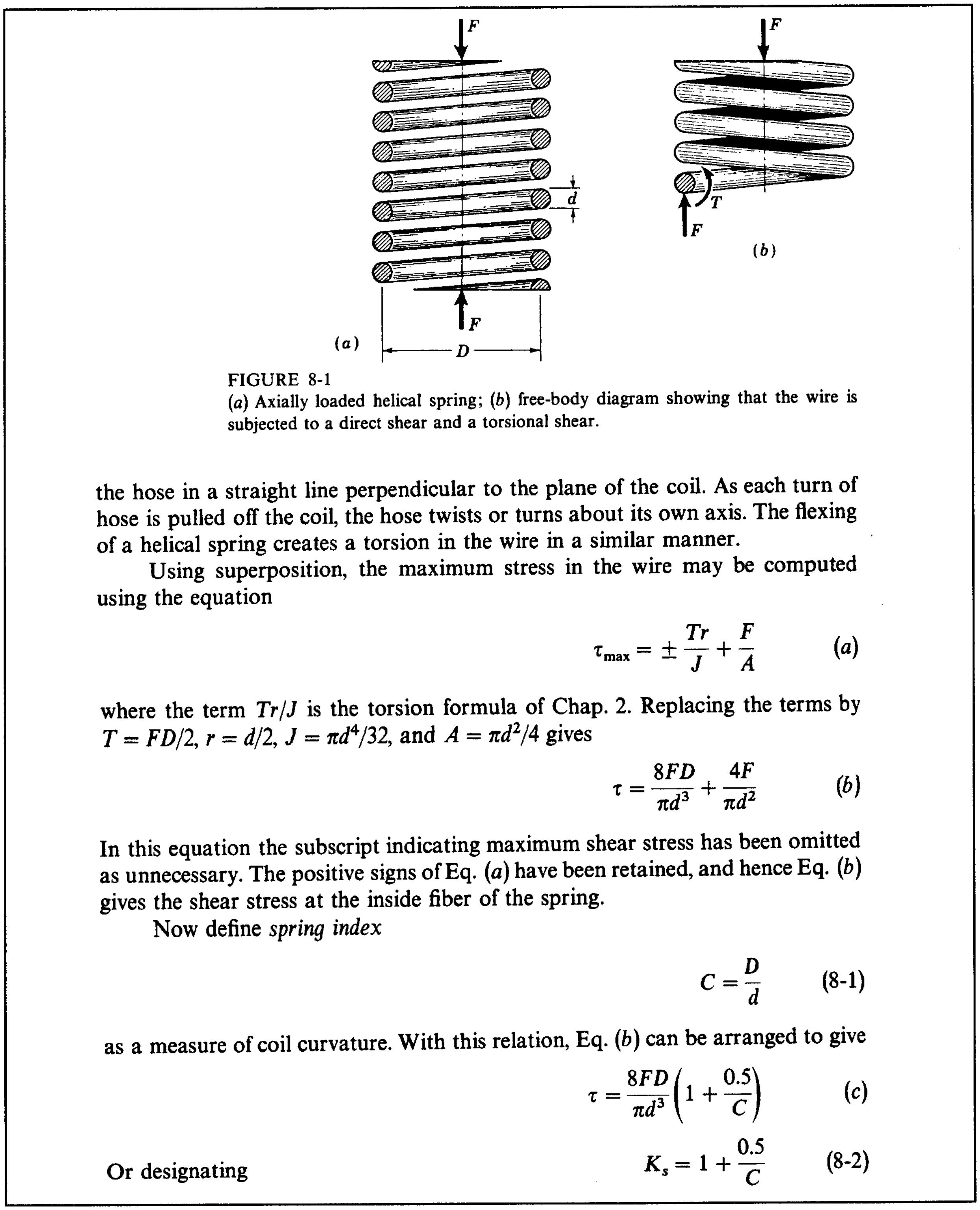
Graphs, tables, and equations captured relationships among sets of variables. Graphs additionally provided a highly visual insight into the mutual dependency of the variables.

Domain specialists read the text and concurrently used the included graphs, tables, and charts. They used note pads to make temporary notes and calculations. Simple calculations were done mentally. More complicated calculations required aids, such as log tables and slide rules. As new ideas, information, and knowledge were generated, the domain specialists captured them in natural forms (such as equations, tables, graphs, and pictures), appended them to text, and communicated the new documents to others in the field. In general, paper-based documents were subject to three principal types of uses: (1) reading and comprehension; (2) interactive calculations using the calculation features (namely, tables, equations, graphs, and pictures) found in the text; and finally, (3) development and recording of new functions (tables, equations, graphs, and pictures). Capturing these essential natural forms and processes in a computer software system is the goal of Natural Computing.

2. A Textbook Example of Calculation Features

Figure 1 shows a sample page from an engineering textbook describing mechanical springs (Shigley, 1977). The page consists of a sketch of a mechanical spring, text, and equations. Figure 2 shows another sample page with a graph, more equations, and more text. By reading the explanations on these pages, an engineer can understand the domain of mechanical springs. By studying the graph on the page, the engineer can understand the trends. At any time, the engineer can obtain values given by the graphs—this is usually called *reading a graph*. While using these pages, the engineer starts with values of D and d, proceeds to calculate the value of the variable C from equation 8-1 in figure 1, and reads the value of the Wahl correction factor K from the graph of spring index versus stress correction factor. This value of K and an input value of F (force) are next substituted into equation 8-4 (fig. 2), and the value of stress τ is calculated. The engineer may next proceed to the page shown in figure 3 and read the values of A and m for a given material from the table 8-2 (fig. 3). These values are substituted into equation 8-10 (fig. 3) for calculating the ultimate strength in tension of the spring material.

This description shows how domain specialists present information in textbooks for others to use to perform calculations. Thus, textbooks are used both to explain the subject and to provide information in the form of text, sketches, equations, graphs, and tables for ready use in calculations. In paper-based documents, there is always more information than a particular calculation requires. A user can perform several alternative sets of calculations by selecting some equations, graphs, and tables and chaining them appropriately for the desired calculation at hand.



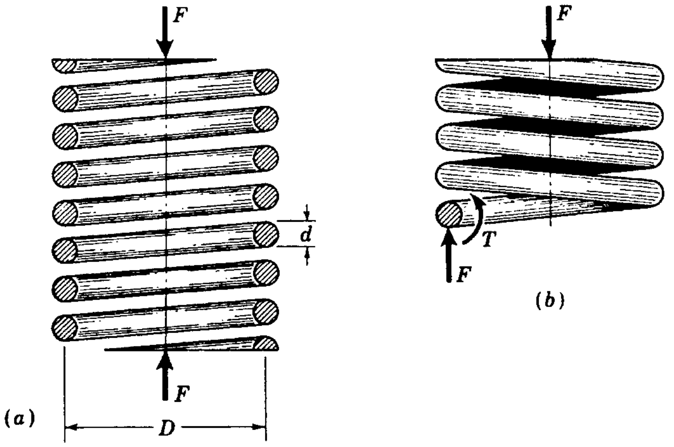


FIGURE 8-1
(*a*) Axially loaded helical spring; (*b*) free-body diagram showing that the wire is subjected to a direct shear and a torsional shear.

the hose in a straight line perpendicular to the plane of the coil. As each turn of hose is pulled off the coil, the hose twists or turns about its own axis. The flexing of a helical spring creates a torsion in the wire in a similar manner.

Using superposition, the maximum stress in the wire may be computed using the equation

$$\tau_{\max} = \pm \frac{Tr}{J} + \frac{F}{A} \qquad (a)$$

where the term Tr/J is the torsion formula of Chap. 2. Replacing the terms by $T = FD/2$, $r = d/2$, $J = \pi d^4/32$, and $A = \pi d^2/4$ gives

$$\tau = \frac{8FD}{\pi d^3} + \frac{4F}{\pi d^2} \qquad (b)$$

In this equation the subscript indicating maximum shear stress has been omitted as unnecessary. The positive signs of Eq. (*a*) have been retained, and hence Eq. (*b*) gives the shear stress at the inside fiber of the spring.

Now define *spring index*

$$C = \frac{D}{d} \qquad (8\text{-}1)$$

as a measure of coil curvature. With this relation, Eq. (*b*) can be arranged to give

$$\tau = \frac{8FD}{\pi d^3}\left(1 + \frac{0.5}{C}\right) \qquad (c)$$

Or designating

$$K_s = 1 + \frac{0.5}{C} \qquad (8\text{-}2)$$

Source: Joseph E. Shigley (1977). *Mechanical Engineering Design*, 3rd ed., McGraw-Hill Book Co., New York, NY.

Figure 1. A sample page containing text, sketch, and equations.

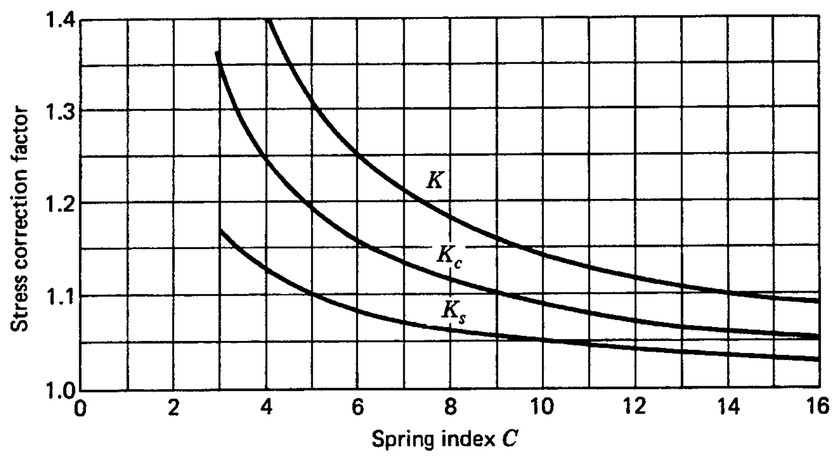


FIGURE 8-2
Values of the stress correction factors for round helical extension or compression springs.

then

$$\tau = K_s \frac{8FD}{\pi d^3} \qquad \text{(8-3)}$$

where K_s is called a *shear-stress multiplication factor*. This factor can be obtained from Fig. 8-2 for the usual values of C. For most springs, C will range from about 6 to 12. Equation (8-3) is quite general and applies for both static and dynamic loads. It gives the maximum shear stress in the wire, and this stress occurs at the inner fiber of the spring.

Many writers present the stress equation as

$$\tau = K \frac{8FD}{\pi d^3} \qquad \text{(8-4)}$$

where K is called the *Wahl correction factor*.* This factor includes the direct shear, together with another effect due to curvature. As shown in Fig. 8-3, curvature of the wire increases the stress on the inside of the spring but decreases it only slightly on the outside. The value of K may be obtained from the equation

$$K = \frac{4C - 1}{4C - 4} + \frac{0.615}{C} \qquad \text{(8-5)}$$

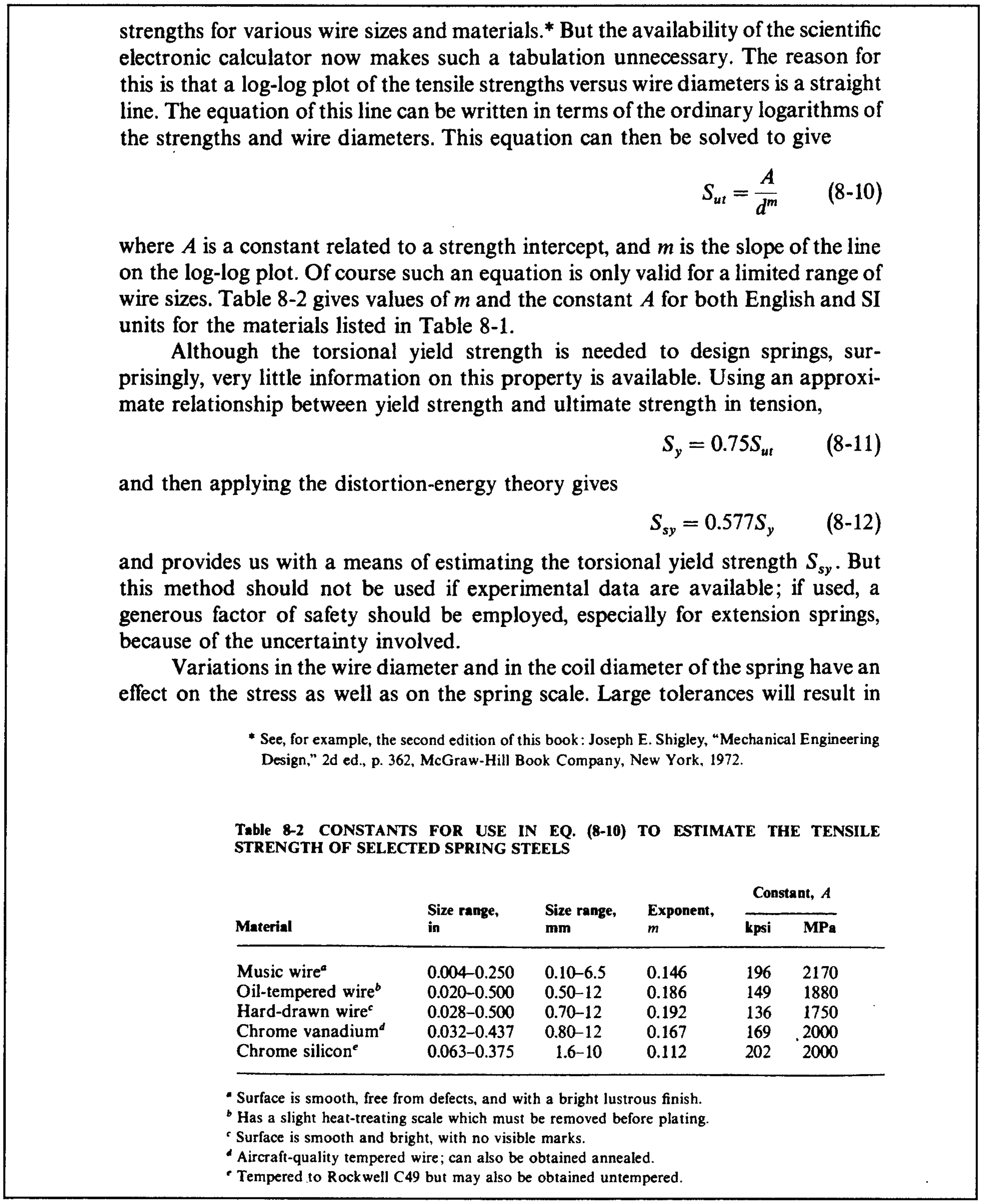
or from Fig. 8-2.

By defining $K = K_c K_s$, where K_c is the effect of curvature alone, we have

$$K_c = \frac{K}{K_s} \qquad \text{(8-6)}$$

Source: Joseph E. Shigley (1977). *Mechanical Engineering Design*, 3rd ed., McGraw-Hill Book Co., New York, NY.

Figure 2. A sample page containing text, graph, and equations.



strengths for various wire sizes and materials.* But the availability of the scientific electronic calculator now makes such a tabulation unnecessary. The reason for this is that a log-log plot of the tensile strengths versus wire diameters is a straight line. The equation of this line can be written in terms of the ordinary logarithms of the strengths and wire diameters. This equation can then be solved to give

$$S_{ut} = \frac{A}{d^m} \qquad (8\text{-}10)$$

where A is a constant related to a strength intercept, and m is the slope of the line on the log-log plot. Of course such an equation is only valid for a limited range of wire sizes. Table 8-2 gives values of m and the constant A for both English and SI units for the materials listed in Table 8-1.

Although the torsional yield strength is needed to design springs, surprisingly, very little information on this property is available. Using an approximate relationship between yield strength and ultimate strength in tension,

$$S_y = 0.75 S_{ut} \qquad (8\text{-}11)$$

and then applying the distortion-energy theory gives

$$S_{sy} = 0.577 S_y \qquad (8\text{-}12)$$

and provides us with a means of estimating the torsional yield strength S_{sy}. But this method should not be used if experimental data are available; if used, a generous factor of safety should be employed, especially for extension springs, because of the uncertainty involved.

Variations in the wire diameter and in the coil diameter of the spring have an effect on the stress as well as on the spring scale. Large tolerances will result in

* See, for example, the second edition of this book: Joseph E. Shigley, "Mechanical Engineering Design," 2d ed., p. 362, McGraw-Hill Book Company, New York, 1972.

Table 8-2 CONSTANTS FOR USE IN EQ. (8-10) TO ESTIMATE THE TENSILE STRENGTH OF SELECTED SPRING STEELS

Material	Size range, in	Size range, mm	Exponent, m	Constant, A	
				kpsi	MPa
Music wire[a]	0.004–0.250	0.10–6.5	0.146	196	2170
Oil-tempered wire[b]	0.020–0.500	0.50–12	0.186	149	1880
Hard-drawn wire[c]	0.028–0.500	0.70–12	0.192	136	1750
Chrome vanadium[d]	0.032–0.437	0.80–12	0.167	169	2000
Chrome silicon[e]	0.063–0.375	1.6–10	0.112	202	2000

[a] Surface is smooth, free from defects, and with a bright lustrous finish.
[b] Has a slight heat-treating scale which must be removed before plating.
[c] Surface is smooth and bright, with no visible marks.
[d] Aircraft-quality tempered wire; can also be obtained annealed.
[e] Tempered to Rockwell C49 but may also be obtained untempered.

Source: Joseph E. Shigley (1977). *Mechanical Engineering Design*, 3rd ed., McGraw-Hill Book Co., New York, NY.

Figure 3. A sample page containing text, table, and equations.

3. Usage of Graphs

Graphs are ubiquitous in books and paper documents of all types. Although graphs have been in use for centuries, research into graphs and their properties is nonexistent. In general, people use graphs quite intuitively and very little instruction is ever provided on graphs. Yet children as young as six years old understand graphs when they see a graph (as shown in figure 4) depicting the trajectory of a baseball or the path of an arrow (Green, 1984). Originating with these simple beginnings, the structure of graphs can become very complex (as I show later in this report). Mature adults learn the structure, properties, and relationships of items in a graph by trial and error, supplemented by intuition.

Figure 5 shows sample graphs with two simple curves. One curve* shows a linear or straight-line relationship between two variables. The x-axis represents time in seconds, and the y-axis shows velocity in feet per second. In the second curve, a curvilinear relationship is shown, where the x-axis represents time in seconds and the y-axis shows distance fallen in feet. A reader can use these curves to find out the velocity or the distance fallen at any time between 0 and 10 s. Figure 6 is a sample of a slightly more complex graph in which the x-axis is represented by a log scale (Gartmann, 1970).

From these sample graphs, the basic functionality of a graph can be discerned. A graph depicts a functional relationship between two variables. Even more importantly, from a calculation point of view, a graph allows a user to read a y-value for a given x-value. In that sense, a graph can be seen as a substitute for an equation or a table; this observation leads one to conclude that a graph represents a mathematical relationship between two (or more) quantities or variables.

We can usually represent a graph as $y = f(x)$, where y is the output value for a given x (input value), with $f(\ldots)$ representing some functional relationship.

Figure 4. A simple graph showing path of an arrow.

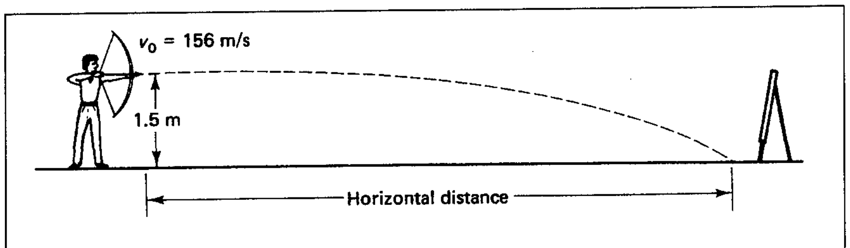


Figure 3-14

Source: C. R. Green (1984). *Technical Physics*, Prentice-Hall, Inc., Englewood-Cliffs, NJ.

*In the literature, terms like *graph*, *curve*, and *line* are used with somewhat similar meanings. In this report, I use the term *graph* mostly to mean a complete graph consisting of a set of curves, the axes, and so on. However, the word *graph* is also used to mean a curve even when one is specifically referring to only one curve. The term *curve* is used to mean one item in a graph that may have many curves. A *line* is the same as a *curve*.

Figure 5. Samples of simple graphs.

Time of fall, *s*	Velocity ft/s; mi/h	Distance fallen, ft
0	0; 0	0
1	32; 21.8	16
2	64; 43.6	64
3	96; 65.4	144
4	128; 87.3	256
5	160; 109	400
6	192; 130	576
7	224; 153	784
8	256; 174	1024
9	288; 196	1296
10	320; 218	1600

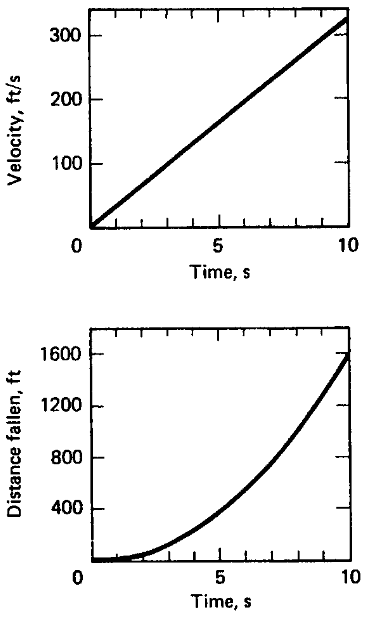


Figure 3-11 Distance fallen and velocity as a function of time. The acceleration of gravity is taken as 32 ft/s^2 and any effect of air resistance is not considered.

Source: C. R. Green (1984). *Technical Physics*, Prentice-Hall, Inc., Englewood-Cliffs, NJ.

Figure 6. A sample graph with a logarithmic scale showing relationship between two variables.

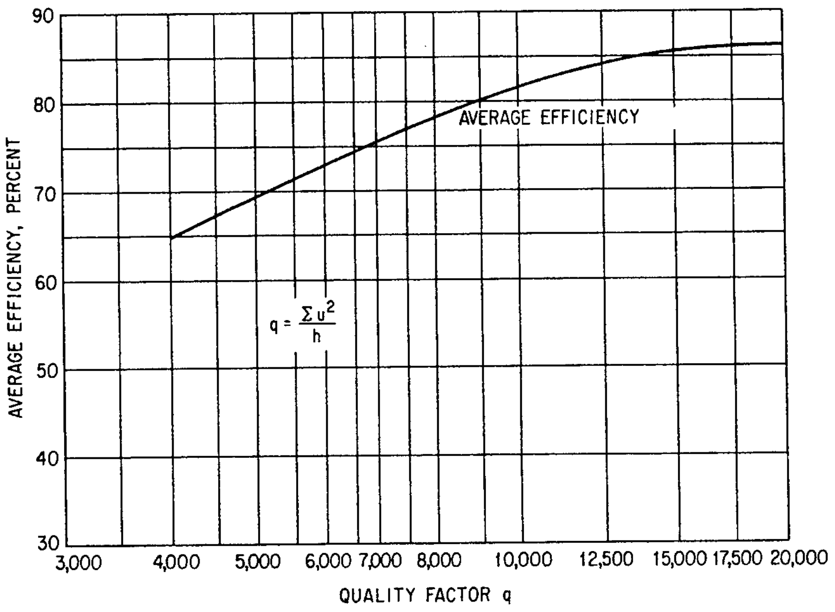


Fig. 5-7 Average efficiency of multistage turbines based on quality factor.

Source: Hans Gartmann (1970). *The DeLaval Engineering Handbook*, 3rd ed., McGraw-Hill Book Co., New York, NY.

Figure 7 shows an interesting sample graph; no curve or line connects the points, because this graph represents a relationship between variables in a discrete domain (Vaughn, 1974). The relationship between the x- and y-values exists only at the points shown on the graph. In between these points, a relationship is nonexistent, invalid, and meaningless. This situation corresponds to tables of data in which interpolation between table values is not permitted.

It is very important to distinguish between graphs that represent data in discrete and continuous domains. Figures 4 to 6 show continuous domains. In a continuous domain, the values between the points on a graph or curve can be obtained by interpolation, while in a discrete domain, such interpolation is not allowed and may be meaningless.

In the days before slide rules, values of reciprocals, sine, cosine, tangent, sinh, cosinh, tanh, and so forth were laboriously calculated and graphs plotted (or tables developed) for later use by scientists and engineers in performing calculations. A graph can be represented best if a large number of points are available. Graphs (a more appropriate term is *lines* or *curves*, as explained earlier in the footnote) are drawn through the calculated points with best estimates (or least squares). Such graphs can be an alternative to tables of data for subsequent calculations. Especially when we are dealing with experimental data, graphs come in handy. As mathematics (particularly algebra) progressed, people preferred to fit equations and use them rather than relying on the plotted graphs or raw tables of data. At any rate, graphs, tables, and equations can be alternative forms for capturing functional relationships between variables in a given problem. Of course, each form (graphs, tables, and equations) has its relative merits, demerits, and suitability for a given application.

Figure 7. A sample graph showing a discrete relationship.

REGRESSION CONTROL CHARTS 85

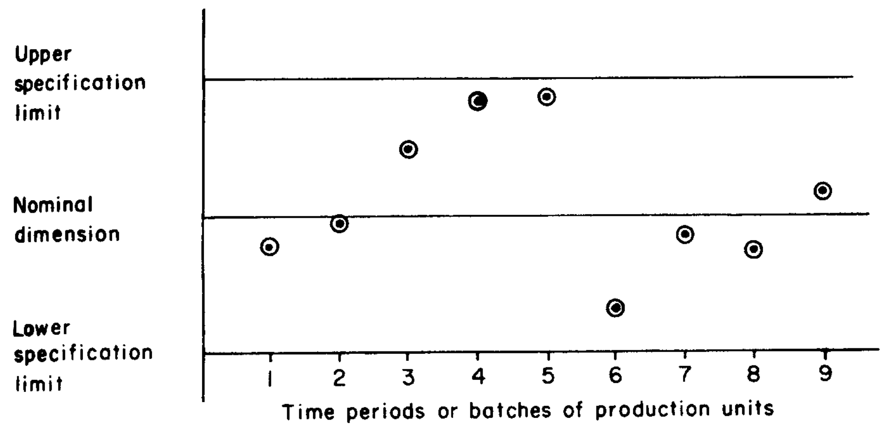


FIG. 6.1. Dimensional trend from tool wear.

Source: Richard C. Vaughn (1974). *Quality Control*, Iowa State University Press, Ames, IA.

Figure 8 shows a sample graph with seven curves (Baumeister and Marks, 1964). In this case, the relationship is between vapor temperature and pressure. Each curve represents this relationship for a different hydrocarbon. Each is labeled with the appropriate name. In using this graph, we begin with two quantities or variables (temperature and the hydrocarbon) and obtain values for the third (pressure). The advantage of a graphical representation is immediately obvious from the graph. A graph gives high visibility to data trends. First, figure 8 shows that the vapor pressure increases with temperature for every one of the hydrocarbons. The shapes of the curves are also somewhat similar. The graph clearly shows which hydrocarbons have very high vapor pressures compared to others for a given temperature. The curves are continuous, thus allowing for interpolation *along* a curve. However, interpolation *between* curves is meaningless, because the types of hydrocarbons are discrete parameters.*

The foregoing samples depict how graphs clearly show the relationships in the data. As the relationships get more complex or as more relationships are presented, graphs become complex, but they are essential to present this insight. For two centuries, thermal engineers dealt with steam charts obtained through laborious experimental work. Steam has a thermodynamic state that is defined in terms of a number of properties, i.e., pressure (p), temperature (T), specific volume (v), internal energy (i), enthalpy (h), and entropy (s). The state is completely obtainable if we know any two of these properties (Keenan et al, 1969). Historically, steam tables and steam charts, also called Mollier charts (fig. 9), have been the main means of representing the thermal states of steam.

Figure 8. A sample graph showing a number of parametric curves.

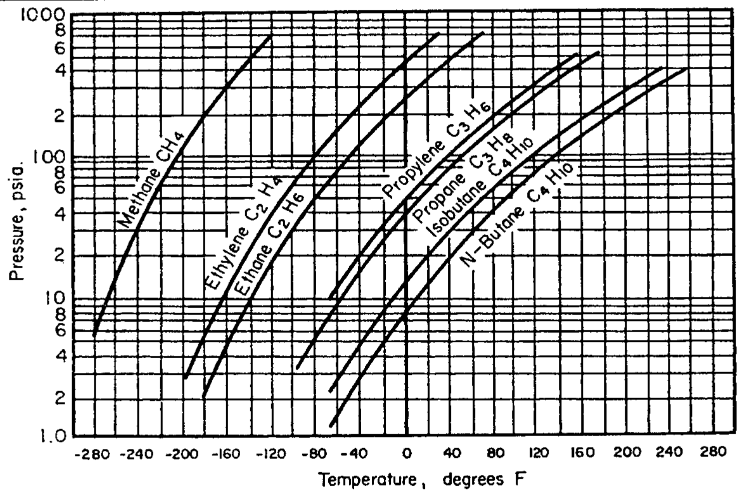
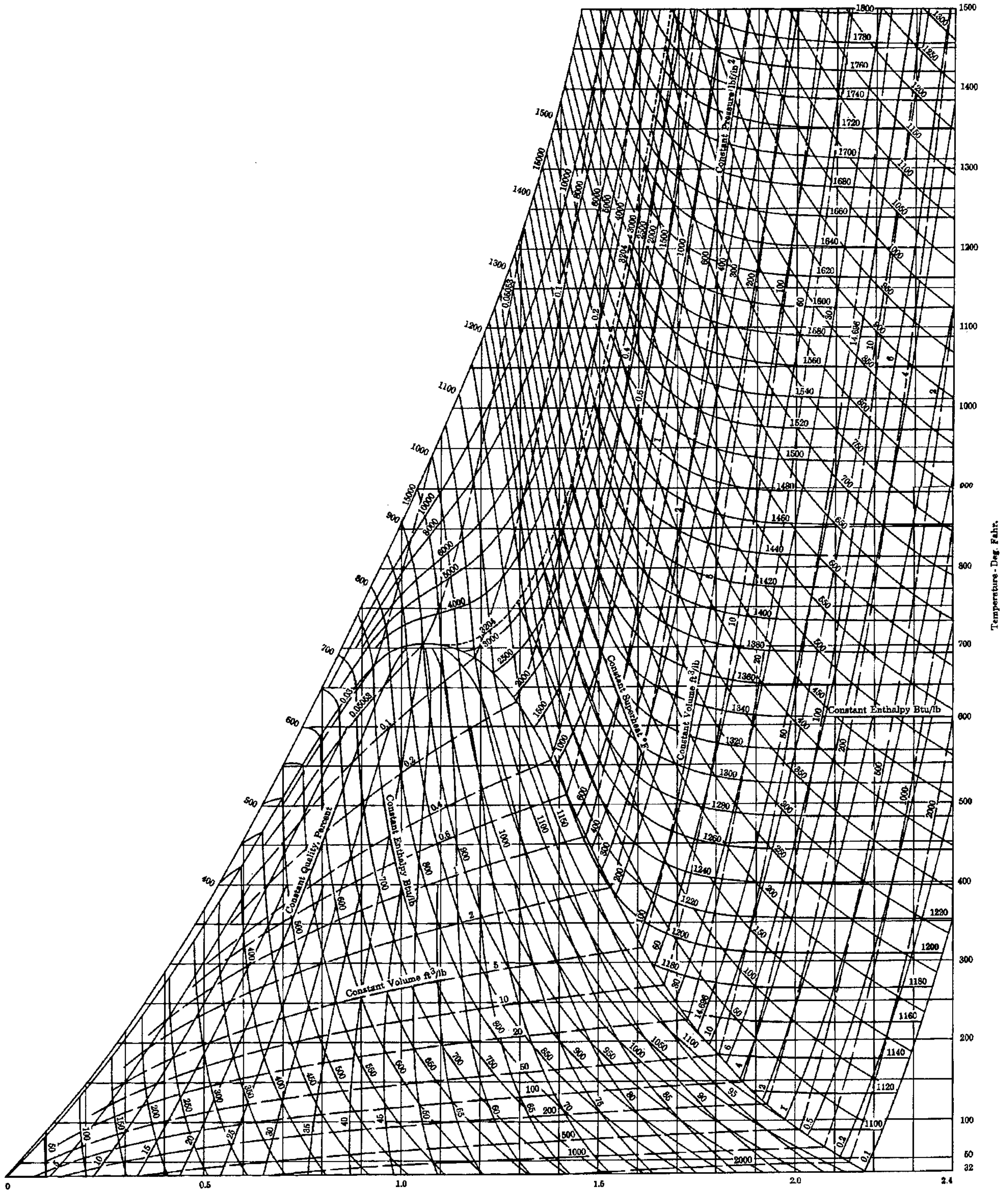


FIG. 21. Vapor pressures of pure hydrocarbons.

Source: Theodore Baumeister and Lionel S. Marks, eds. (1964). *Mechanical Engineers' Handbook*, 7th ed., McGraw-Hill Book Co., New York, NY.

*The term *parameter* represents a variable other than the two primary variables represented on the x-axis and the y-axis. In a two-dimensional graph representation of data of the form $y = f(x, p_1, p_2, \ldots)$, $p_1, p_2, \ldots$ are the parameters. Sometimes, these parameters are also called contours or isolines.



Source: Joseph H. Keenan, Frederick G. Keyes, Philip G. Hill, and Joan G. Moore (1969). *Steam Tables: Thermodynamic Properties of Water Including Vapor, Liquid, and Solid Phases,* John Wiley and Sons Inc., New York, NY.

Figure 9. A sample of a complex graph: Mollier chart showing properties of steam.

However, with the advent of high-performance digital computers, complex functional (algebraic) relationships or equations between these properties have been developed. A number of computer programs are now available that facilitate calculations for a steam plant operation without the use of the traditional steam tables or steam charts (Mollier charts). Although these computer programs can be used to obtain the properties of steam, they cannot match the visibility and intuition provided by the steam tables and the steam charts.

Figure 10 gives a sample of a highly visible presentation of the operation of a steam plant by means of a schematic diagram and a temperature-entropy (*T-s*) diagram (Van Wylen and Sonntag, 1965). The sketch and the graph together show all the components that make up the system and the representation of the states of the steam as it flows through the four components: boiler, turbine, condenser, and pump. Specific functional relationships apply to these states. Historically, the states (i.e., the property values) of the steam were calculated from the Mollier charts. The accuracy of these charts is limited by their scale. Steam tables were used instead to improve the accuracy of the calculations (at the expense of some visibility). With the advent of digital computers, software was developed to represent the thermal behavior of the steam. Unfortunately, such software is both complex and opaque, and its use in steam plant calculations made problem solving a "black box" process. The aim of Natural Computing is to restore to software programs this visibility feature while preserving the accuracy and efficiency of computers.

Figure 10. Simple steam power plant schematic and temperature-entropy diagram.

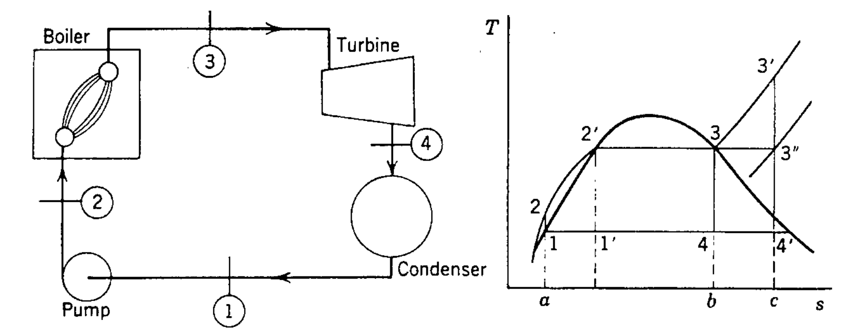


Fig. 9.1 Simple steam power plant which operates on the Rankine cycle.

Source: Gordon J. Van Wylen and Richard E. Sonntag (1965). *Fundamentals of Classical Thermodynamics*, John Wiley and Sons Inc., New York, NY.

4. Ends and Means

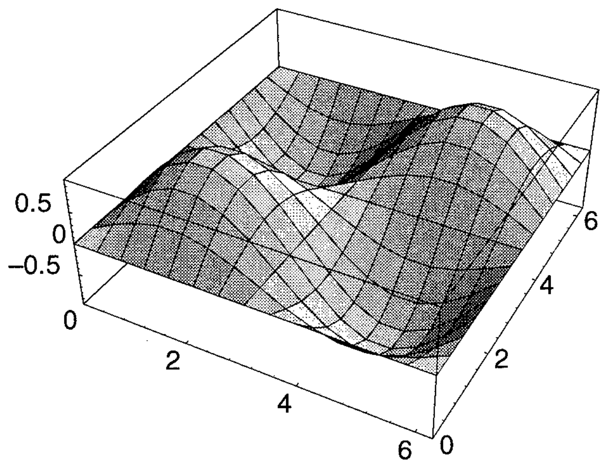
In the last decade or so, traditional software has succeeded in representing and generating tables and graphs. Text, graphs, equations, and pictures coexist in paper-based documents. Even so, these objects are generated as ends (display or printed documents) and not as means for further continual computing. I will elaborate this distinction in the following.

Computer programs are now readily available that generate graphs in vivid colors and multiple dimensions. Figure 11 is a graph generated by Mathematica® software (Wolfram, 1991). While this graph looks excellent, it is meant to be displayed on a computer screen, but it is not meant to be used by another program in a continuum of calculations.

Coad and Yourdon (1991) caution software practitioners that if the application of a software engineering "method" produces a monument of paper, then something is wrong—either in the method, in the application of the method, or perhaps both. They lament, "if we lose sight of people and begin producing charts, diagrams, and piles of paper as *ends* [italics added] unto themselves, we fail to effectively communicate." Currently, most software methods generate graphs that can be used as records and not as dynamic relationships. Contrast this with the traditional use of graphs in books: A scientist can readily use a graph from a book (on paper) either to look at or to act as a relation in his or her calculation. But with the traditional computer software, the reuse of graphs is limited and circuitous, since computer tools are not presently available that treat graphs as relationships.

Figure 11. An example graph generated by Mathematica®, useful for display only.

```
In[4]:=
    Plot3D[Sin[x] Sin[y], {x, 0, 2Pi}, {y, 0, 2Pi}]
```



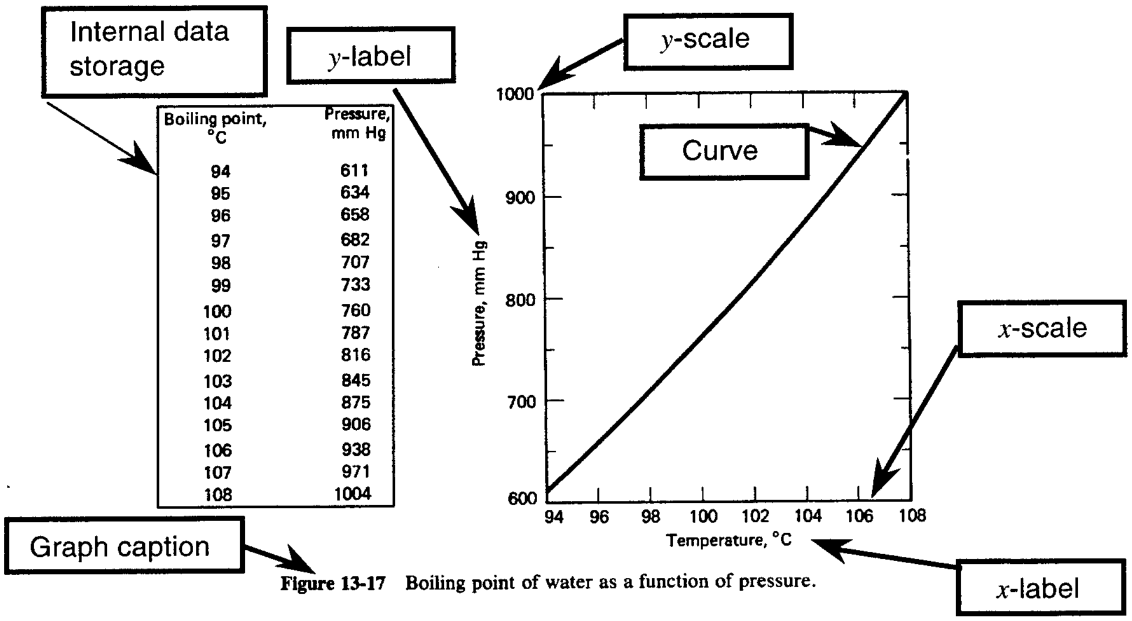
In an earlier report (Karamchetty, 2000a), I demonstrated that a functional relationship between a set of variables can be represented by an equation, a graph, a table, or a computer program. I also showed that graphs provide the greatest insight to a user. By developing data structures and methods to represent graphs as means, as well as ends, we will be able to facilitate their seamless reuse in computer systems. Graphs will then become computational means for communication among people, understanding by people, and further generation of knowledge by people. Graphs could then be copied and pasted into new computer programs as part of a program development process.

5. Informal Survey and Analysis of Practical Graphs

In this section, I provide a number of examples of graphs from textbooks, handbooks, brochures, and newspapers to demonstrate the richness and complexity of data, information, and functional relationship representations in practical graphs. These examples allow me to identify and describe characteristics and properties of graphs.

As described in section 3, a graph represents some characteristics and values in a domain. A graph is usually displayed as a two-dimensional curve. A graph contains a set of characteristics (also called quantities or variables) that are represented along the abscissa and the ordinate. These names of two characteristics are usually used as labels for the *x*- and *y*-axes. The value sets are the coordinates of a point on the curve. Since the values are usually not measured in terms of the physical dimensions, they are interpreted by means of the *x*- and *y*-scales. Where there is more than one curve on the graph, parameter names (and values) are used to identify distinct curves. A specific graph can be identified by a caption that gives it a unique identity in the document. In documents, such as books, we find a list of graphs that brings together all the graph captions to one location (generally with the table of contents listing the appropriate page number). Thus, a simple graph will consist of a graph caption, *x*- and *y*-labels, a list of parameters, and *x*- and *y*-scales. The graph caption is a string. The *x*- and *y*-labels and the parameters are all strings. In some instances, these strings also contain the units of the values.

Figure 12 is a sample of a simple graph with its parts identified. Note that the graph caption, the *x*- and *y*-labels, and the *x*- and *y*-scales are combinations of numbers and strings that can be represented in the computer in conventional ways. However, the curve itself is a line that has no direct computer representation. A table of values can be used to represent a curve in a computer. In figure 12, the table of data labeled *internal data storage* can be used as a representation for the curve. The curve is usually drawn from a table of data, which can be stored in the computer. In other instances, a curve may be drawn by use of an equation, which can also be stored in the computer. In a third type of situation, where a curve is obtained as a result of a long procedure, a data table may be generated or an equation fitted to represent the curve (many sophisticated curve-fitting programs are available to satisfy this need). The important point here is that a curve is stored internally in the computer as a table or an equation or a procedure. If we consider the internal data storage table in figure 12, we can see that a table with a number of columns could be used to develop the data in figure 13 (Green, 1984).

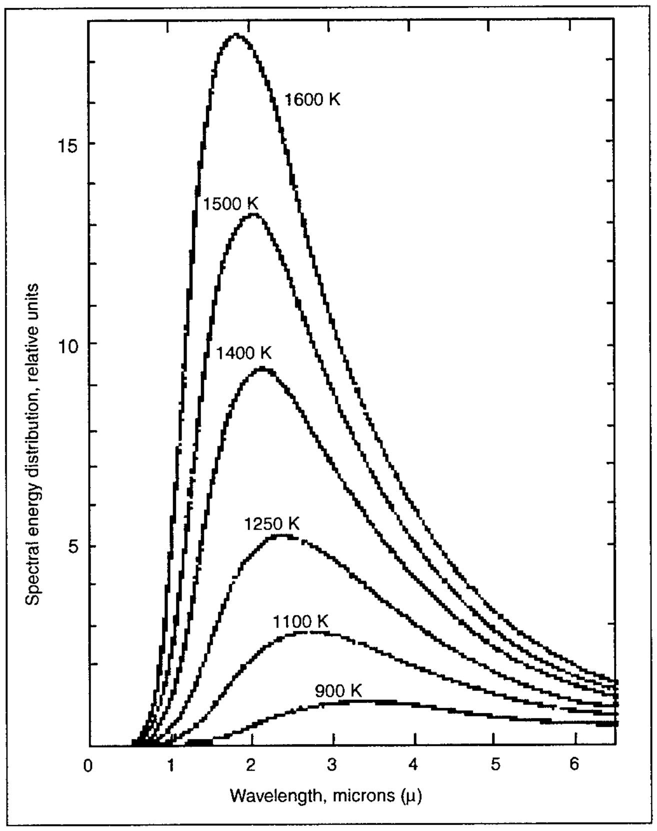


Boiling point, °C	Pressure, mm Hg
94	611
95	634
96	658
97	682
98	707
99	733
100	760
101	787
102	816
103	845
104	875
105	906
106	938
107	971
108	1004

Figure 13-17 Boiling point of water as a function of pressure.

Figure 12. An anatomy of a simple graph.

Figure 13. A sample graph with several parametric curves.



Source: C. R. Green (1984). *Technical Physics*, Prentice-Hall, Inc., Englewood-Cliffs, NJ.

In figure 8, given in section 3, the y-axis is not a linear scale but a logarithmic scale. In contrast, both x- and y-scales in figure 14 are logarithmic (Baumeister and Marks, 1958). When the scales are logarithmic, the cursor position on the curve requires care in scaling. Special interpolation processes are needed to preserve the accuracy of calculations involving logarithmic scales.

Figure 15 shows a sample graph containing a picture. Additionally, the graph also has text, which guides the reader toward "blunt fillets" and "sharp fillets." In this example, the sketch and the text do not affect the calculations with the curves, but they improve the understandability of the information in the graph.

Figure 16 shows how complex graphs can become (Karassik et al, 1976). This sample graph is rich with information that will be concealed if we replace it with a computer program. The graph shows three regions: the laminar region, the transition region, and the turbulent region. In the laminar region, the y-value depends only on the x-value. In the turbulent region, the y-value varies very little or only slightly with the x-value, but is dependent on the value of the parameter (the pipe roughness). When a hydraulic engineer does calculations and makes design choices, the region in which the flow is operating should be known. With a graph like that in figure 16, the region of operation is fully and readily evident. The use of a computer program in place of this graph would seriously sacrifice such problem visibility. This example clearly shows that a graph contains a lot of information and illuminates the problem. The user's attention is drawn to the opportunities and pitfalls in the problem.

Figure 14. A sample graph with logarithmic scales on both axes.

THERMODYNAMICS OF FLOW OF COMPRESSIBLE FLUIDS 4–67

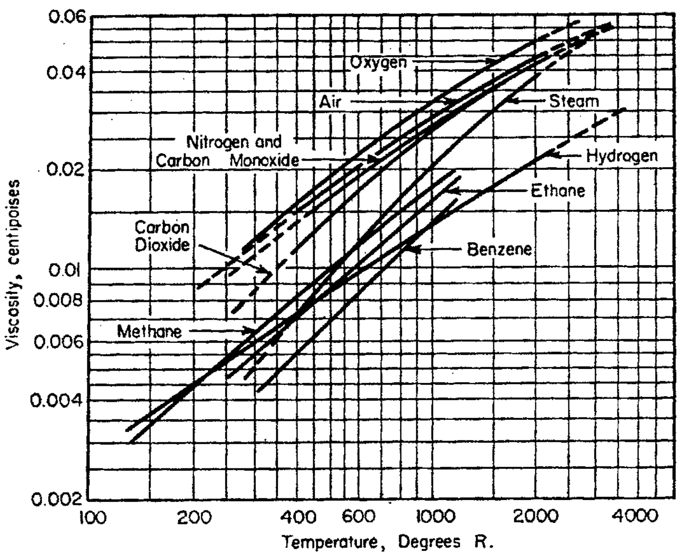


FIG. 38. Viscosity of gases.

Source: Theodore Baumeister and Lionel S. Marks (1964). *Mechanical Engineers' Handbook,* 7th ed., McGraw-Hill Book Co., New York, NY.

One could continue the survey and discover many other graph features in books. However, these examples are sufficient to draw the reader's attention to the basic characteristics involved in computer representation of graphs.

Figure 15. A sample graph containing a picture.

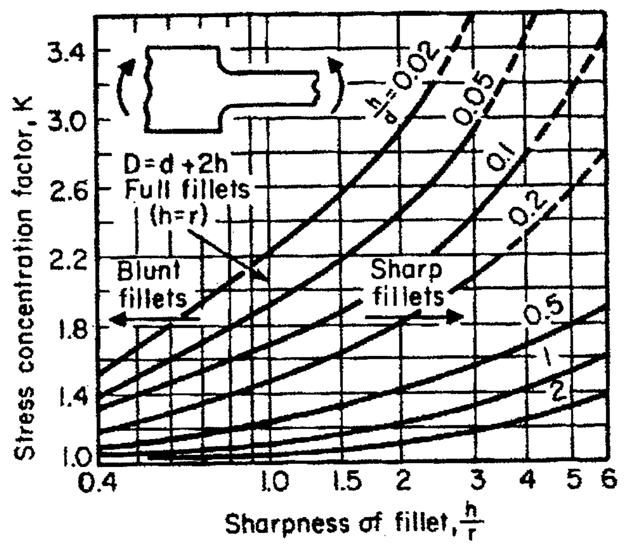


FIG. 11. Flat plate with fillets, in bending.

Source: Theodore Baumeister and Lionel S. Marks (1964). *Mechanical Engineers' Handbook*, 7th ed., McGraw-Hill Book Co., New York, NY.

Figure 16. A sample of a complex graph.

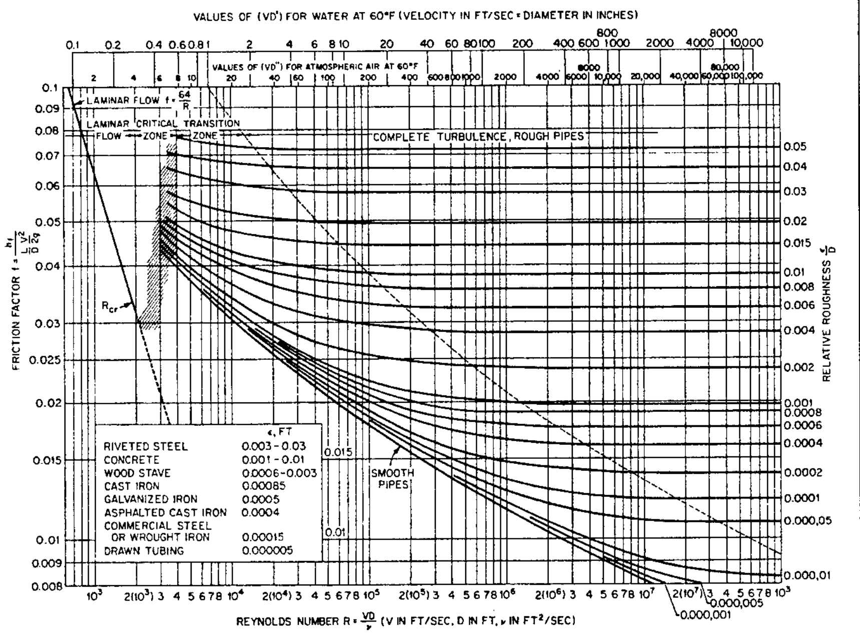


Fig. 28 Moody diagram. (V. L. Streeter, "Fluid Mechanics," 5th ed. Copyright 1971 by McGraw-Hill Book Company, New York)

Source: Igor J. Karassik, William C. Krutzsch, Warren H. Fraser, and Joseph P. Messina (1976). *Pump Handbook*, McGraw-Hill Book Co., New York, NY.

6. Computer Representation and Usage of Graphs in Natural Computing

Although most modern word processors allow graphs to be imported as displays, not one supports the use of graphs as functional relationships. In contrast, the focus in Natural Computing is on the *use* of graphs from the perspective of calculations rather than display. Therefore, this section emphasizes the representation of a graph as part of a document from the point of view of a calculation.

6.1 Structure of a Graph

The survey in section 5 of practical graphs found in books, newspapers, and brochures suggests a minimal structure for adopting such graphs for computer applications. The anatomy of a simple graph, shown in figure 12, consists of a graph caption, x- and y-labels and parameter labels, x- and y-scales, types of scales, and a table or equation for storing data internally.

In general, these and other attributes of a graph can be programmed into a graph class. The simple graph anatomy may form a base class, while other complex features can be programmed into derived classes. A basic class constructor can be designed to create a blank graph. A destructor will delete the object and its components and release the memory space back to the system. A number of operations on graphs can be programmed into these classes. First, a set of filler operators can be developed to perform functions connected with filling a (blank) graph with data. A set of editor operators can be developed to edit and modify the data in a graph. Display operators can be designed to display a graph on a monitor, or print it on paper, whether as a stand-alone display or as part of another object. Storage operators can store persistent data that can be used to reconstruct a graph at any time during its life. Interactive calculation operators provide the capability for the graph to be used to provide results in the interactive mode. The program calculation operators allow the graph to be embedded into a procedure for use by that procedure. Overloaded arithmetic operators allow the graph to perform functions on itself. Examples of such operators include a "+" operator to add two similar graphs to create a third graph; a "–" operator that subtracts a graph from another (compatible graph) to create a third graph; a "*" operator that performs a scalar multiplication of a graph; and a "/" operator that performs a scalar division of a graph. A simple class example is shown in figure 17.

Other sets of operations can be developed that would make graphs very effective in programming. For example, consider the seven curves in figure 8. In a particular application, a specialist may wish to eliminate some of the curves and leave only a subset of them. This can be done if an operator is available to the graph class that allows for the deletion of parametric curves and the creation of a graph that is a subset of a given graph. In reverse, by using an insertion operator, we may wish to add a

Figure 17. A typical example of a simple graph class.

```
// ----- A simple graph class
class Graph {
private:
    string caption;
    int nl=1;
    labels x_label, y_label, p_label[nl];
    scale x_scale, y_scale;
    scale_type x_scale_type, y_scale_type;
    Column_header column_header ;
    Row_header row_header;
    Data_body data_body;
    bool Interpolatability;
public:
//  constructor function(s)
    Graph() {}

// destructor function(s)
    ~Graph() {}

// Graph functions
    Graph_blank();
    Graph_edit();

// Overloaded operators
    Graph operator+ (Graph graph1, Graph graph2);
    Graph operator+ (Graph graph1, Graph graph2);
    Graph operator* (Graph graph1, int mult);

// superset and subset functions
    Graph supergraph (Graph graph1, Graph graph2);
    Graph subgraph (Graph graph, int n, ...);
// Other functions ...
};
```

new curve to an existing set of curves. The graphs still represent the same x- and y-labels, but the list of parameters and the values are changed. Such operations allow for the creation of superset graphs and subset graphs. Figure 18 illustrates the creation of a superset graph that combines two curves with temperature values of 100 °F and 200 °F as parameter values. Given the data structure of the graph class, one can obtain this superset graph by adding the list of volume values corresponding to a list of pressure values with a new parameter with a value of (temperature equal to) 200 °F.

Mere addition or deletion of curves in a graph is thus straightforward. One can also develop graph operations that perform arithmetic manipulations on graphs. For example, a graph containing two curves, *month vs income* and *month vs expenditure*, can be manipulated to yield a *month vs profit* graph, if another curve is defined called *profit*, which is the difference between the income curve and the expenditure curve (see fig. 19). Again, because of the data structure of the graph class, one can obtain the result graph by generating a list of *profit* values through subtracting the *expenditure* list values from the *income* list values. Thus, operations on

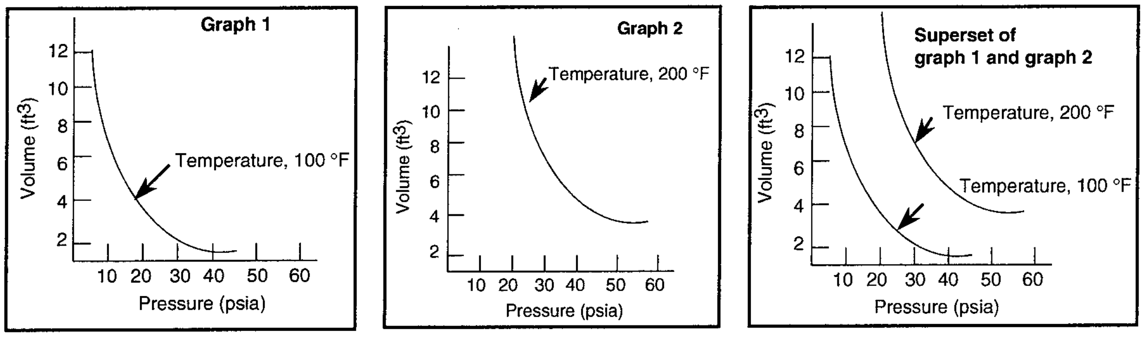


Figure 18. An example of creation of a superset graph.

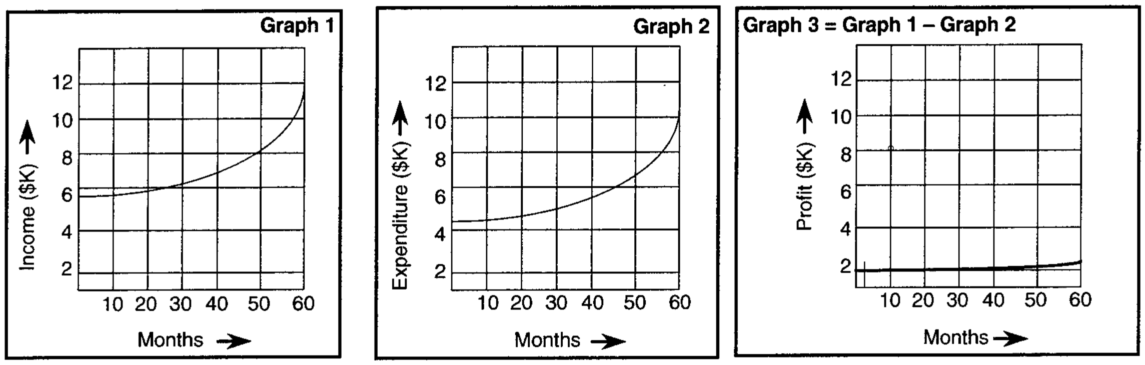


Figure 19. Example showing an arithmetic operation on graphs to generate another graph.

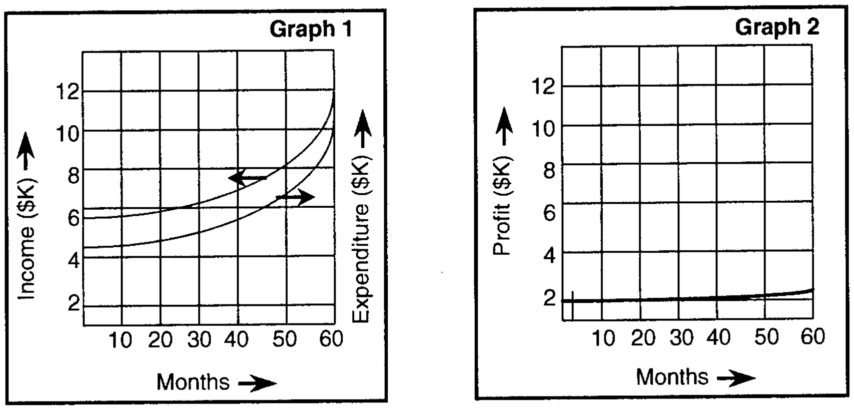
graphs should be able to generate new graphs by addition and/or deletion of curves and by arithmetic manipulation. We can achieve this result by simply writing a functional relationship:

result_graph = graph_1 + graph_2.

If the income and expenditure curves are already part of a single graph, one can obtain the result graph by subtracting the corresponding curves on the same graph as shown in figure 20. Mathematical operations on graphs make graphs very valuable and highly useful. Subtraction, scalar multiplication, and division can constitute other operations on graphs. More complex operations can be defined by combinations of these basic operations.

Some operators in a graph class work to endow features like "responsibility" and "justification" to a graph object. For example, a graph class can check for consistency of units. In the calculation mode, the graph can check the units of the variables in the graph data and the units of the variables in the query. The justification operators can act to adjust the units, thus ensuring that unit errors do not occur with Natural Computing objects.

Figure 20. Example showing an arithmetic operation on a set of curves in a graph to generate another graph.



Graph 2 = income curve – expenditure curve

The graph class can possess this capability because like all Natural Computing features, graphs know that variables carry values and units. In general, all calculation features in Natural Computing work with a set of four items (characteristics): (1) a name of the item, (2) a variable representing the item (called notation), (3) its value, and (4) its units. The characteristic belongs to or is an attribute of a system under discussion or calculation. A variable is a symbol chosen to represent the physical quantity or characteristic for brevity in writing relationships or equations. The characteristic has a definite value at a given state. Its value is given in some physical units. (In many documents, the relationships between the variables and their descriptions are often given in lists of notation or lists of symbols.)

In the graph class, these four items are used in the following general ways. The *x*- or *y*-label of a graph contains a variable and/or its description. The units for a quantity are either given in the graph data or are obtainable from the notation. Where the units are given in both the graph data and the notation, they should be checked for compatibility. For example, a quantity like velocity can have units of *feet per second* and *miles per hour*. Ensuring that they are either the same or are compatible is a responsibility feature of a graph object. Converting them to a common set of units is a justification feature of a graph. When a graph is queried, the units in the query (input) and the units for the quantity in the graph should be compatible. Likewise, there must be compatibility between the units of a graph quantity and the units of the output quantity. By treating units and conversions simultaneously with functional manipulation in Natural Computing, one avoids the unit-related errors that have been the cause for much grief in traditional computer programs.

Computer (software) system development and domain (knowledge) development are both intricate processes. System development consists in the development of the graph classes with operations in those classes. These classes (along with other Natural Computing objects) will be made available in the form of a Natural Computing software tool kit to domain specialists who can incorporate their graph information into the graph objects wherever they occur in their problem domain.

Use of a Natural Computing tool kit will allow a variety of domain specialists to incorporate their domain information into software, and robust domain growth will take place over time. As this process continues, domain specialists will find that certain graph features are not available in a version of the Natural Computing software tool kit. Computer software developers can then step in and enhance the capabilities of graphs by adding new classes that will accommodate new graph attributes and new graph operators. This new tool kit will allow representation of more extensive domains. Thus, growth in domains and in the Natural Computing software tool kit will occur iteratively (see fig. 21). As domain specialists encounter or invent new graph features (structure or behavior), software developers will play a primary role in developing extended graph features. But once the tools are available, domain specialists will take over and program the calculation procedures in their domains.

6.2 Choice and Use of Graphs

Just as word processors contain document manipulation tools and graphics packages contain picture manipulation tools, a Natural Computing software tool kit contains several graph tools for use by domain specialists. One will select a graph template that best fits the needs of one's application. Then, an instance of a blank graph is created. In the creating/editing mode, all data and information are entered into a graph. The domain specialists are responsible for filling in the footnotes and other notes as appropriate. The graph object automatically develops a number of behavioral characteristics and presents them for the domain specialist's review. For example, the limit values (minimum, maximum, and singularities) of a curve are recorded automatically (minimum and maximum values of a variable prevent a naive end user from extrapolating a graph outside its allowable bounds). The graph object uses these limits to flag an error message when a user tries unallowable values. This feature was inspired by human usage of graphs. Such checks, which are often provided by domain specialists in real-world applications, are now given to the graph classes as responsibilities.

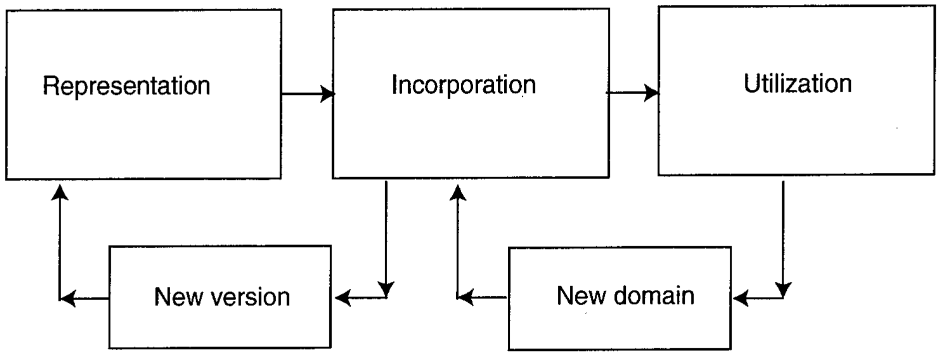


Figure 21. Iterative cycle of graph development in Natural Computing system.

A graph object will also have a set of input (or query) templates and output (result) templates (see fig. 22). Since a graph is a functional relationship between quantities (characteristics) identified on the x- and y-axes and the parameters, given one set of these quantities, all others can be determined from the graph (in Natural Computing). One can use this functional relationship in generating query templates. The user can choose the appropriate input template and type in a value for the independent characteristic or variable and submit it to the graph object. The template also guides the user with the limits on the variable values. These guidance values in the templates protect a graph from invalid or out-of-range queries. A graph will return a result by means of the output templates. A graph with input and output templates can be likened to a hardware component, with its input and output sockets. These input and output templates are used to connect different functional objects into a procedure or program (see fig. 23).

Since the curves on a graph are represented by tables or equations, using a graph for results is very simple. A query consists of an x-axis variable-value pair and a parameter value. The graph object returns the value(s) corresponding to the y-axis variable.

6.3 Insight into Graph Data

Educators criticize current software systems as "black boxes" because the solution method is incomprehensible from the software. The user learns nothing from using the software. For that matter, even the domain specialists do not understand what is in the code once their domain information (actually, algorithms shorn of all explanatory information) is put into code by a software developer. With traditional media (paper, calculator, and pencil), a student's learning improves proportionately with the number of problems solved. In contrast, with traditional computer software, a student's learning does not improve with the number of problems solved. Graphs in Natural Computing, as described in this report, preserve explanatory information and show functional relationships between variables. A student can, therefore, realize opportunities available and watch out for pitfalls in the problem domain represented by each graph. Figure 16 (sect. 5) is an example of a graph with opportunities that a user will understand from the display.

6.4 Testing a Graph in Isolation

Testing is a key task in both software development and application development. In Natural Computing, a domain specialist can test each graph in isolation for a variety of inputs. Owing to the built-in justifications, limits, behaviors, and responsibilities, a Natural Computing graph class can be effective in eliminating software and application bugs. By localizing and fixing errors, one can prevent their propagation to other objects to cause a chain reaction of errors. Isolated testing is a major step in very nearly guaranteeing the correctness of an entire application.

Figure 22. Interactive use of a graph along with input and output boxes.

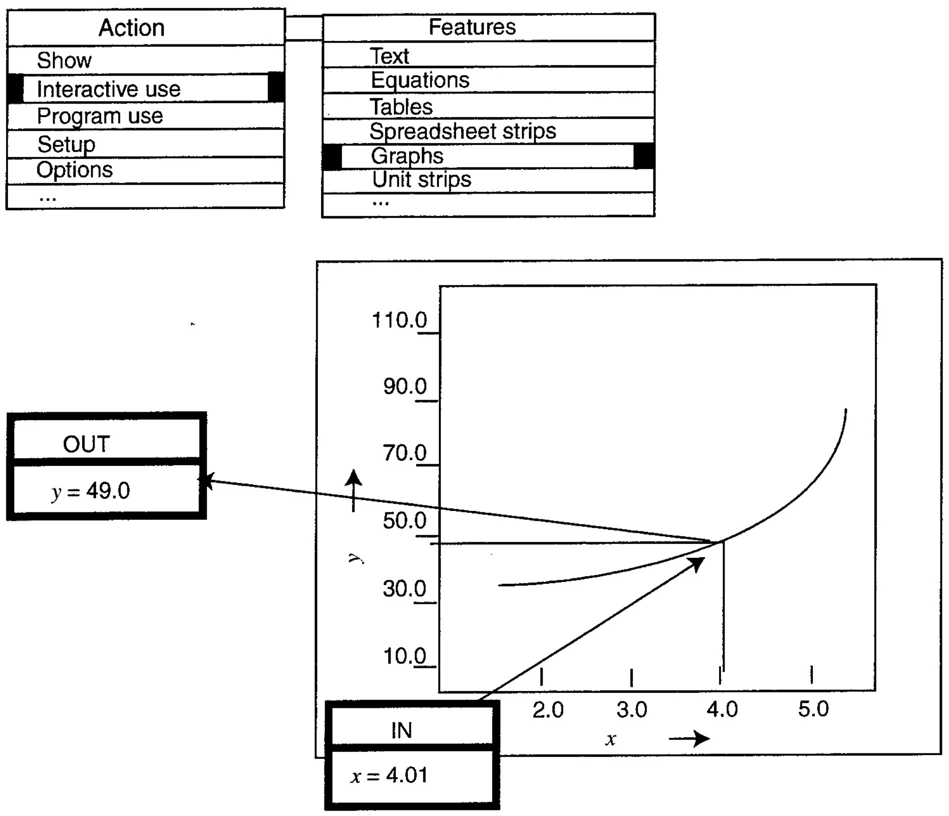
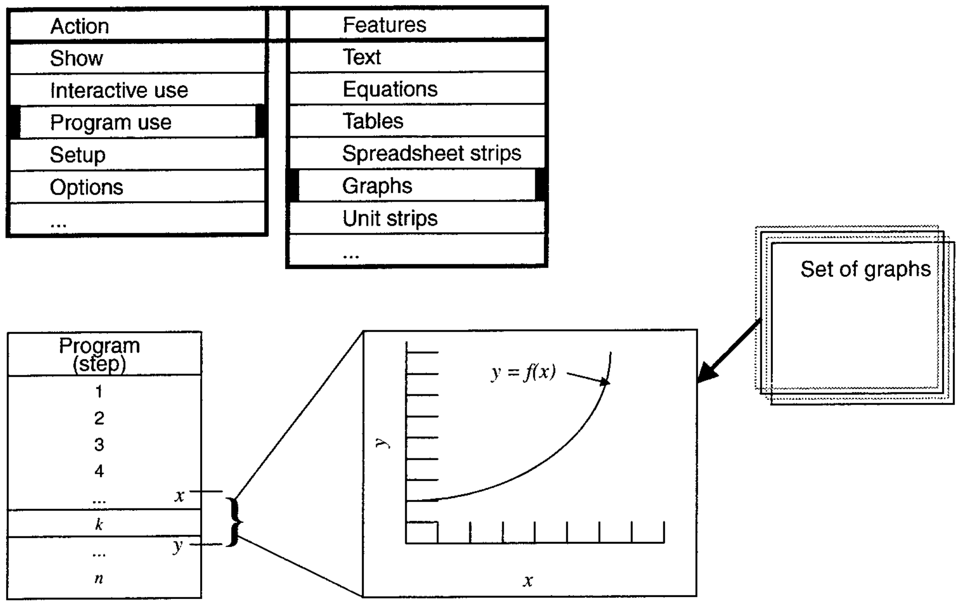


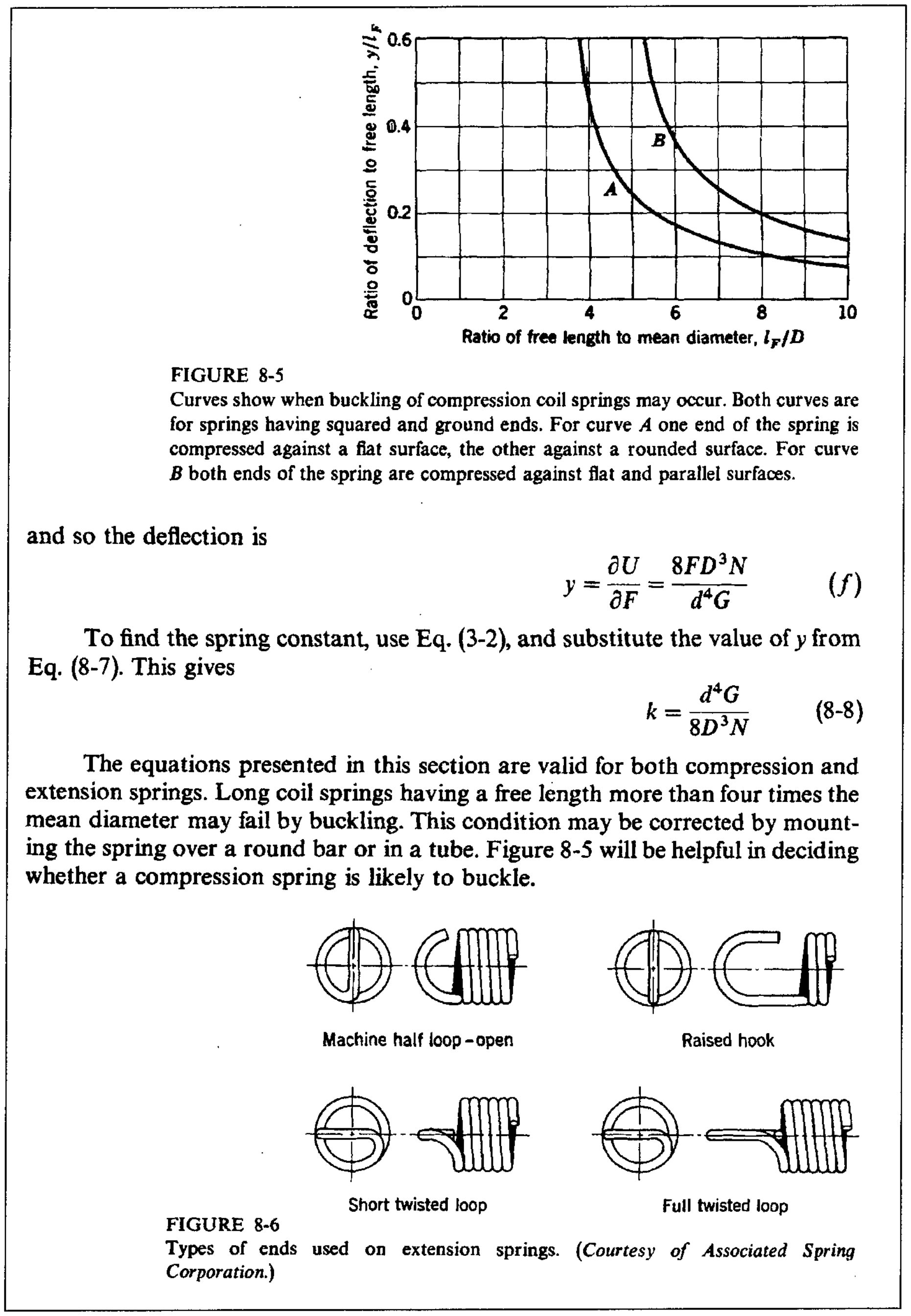
Figure 23. Program or procedure building using a graph.



6.5 Embedding Graphs in Text

Just as in textbooks and other paper documents, Natural Computing graphs are embedded in text, similar to the sample in figure 24 (Shigley, 1977). Readers can read a document and get a general idea of the information presented there, or they can choose to use graphs interactively by activating them. Such an interactive facility is useful for studying a graph and understanding trends in the domain. While developing a procedure or a program, a user can *copy and paste* a graph from the text into the procedure. The display aspects of a graph are seen in the graph object embedded in the text, while the calculation aspects of a graph object are accessible to the procedure object. Once set up into a procedure, a graph object is available to that procedure, which itself can be saved for subsequent use. As described in the report on Natural Computing (Karamchetty, 2000a), one can develop complex procedures by connecting several graphs, equations, tables, and procedures. Any and all Natural Computing objects can be embedded in a Natural Computing document object and can be activated by a user as needed.

Figure 24. Text with embedded graph.



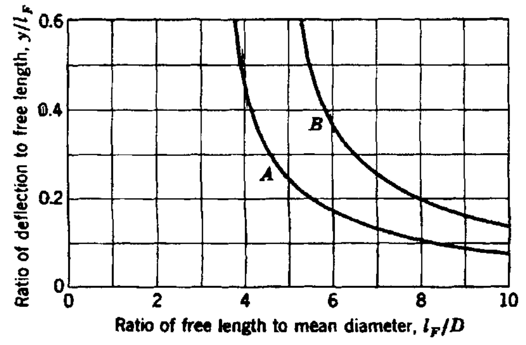

Ratio of deflection to free length, y/l_F
0.6
0.4
0.2
0
0
2
4
6
8
10
Ratio of free length to mean diameter, l_F/D
A
B

FIGURE 8-5
Curves show when buckling of compression coil springs may occur. Both curves are for springs having squared and ground ends. For curve *A* one end of the spring is compressed against a flat surface, the other against a rounded surface. For curve *B* both ends of the spring are compressed against flat and parallel surfaces.

and so the deflection is

$$y = \frac{\partial U}{\partial F} = \frac{8FD^3N}{d^4G} \qquad (f)$$

To find the spring constant, use Eq. (3-2), and substitute the value of *y* from Eq. (8-7). This gives

$$k = \frac{d^4G}{8D^3N} \qquad (8\text{-}8)$$

The equations presented in this section are valid for both compression and extension springs. Long coil springs having a free length more than four times the mean diameter may fail by buckling. This condition may be corrected by mounting the spring over a round bar or in a tube. Figure 8-5 will be helpful in deciding whether a compression spring is likely to buckle.

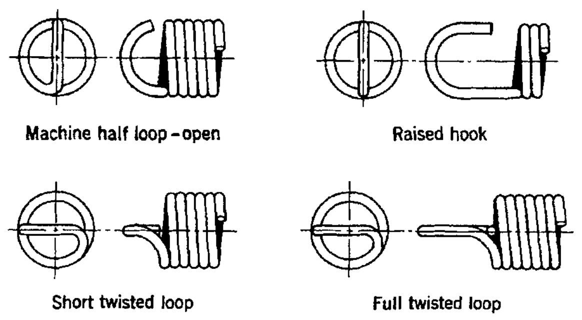


FIGURE 8-6
Types of ends used on extension springs. (*Courtesy of Associated Spring Corporation.*)

Source: Joseph E. Shigley (1977). *Mechanical Engineering Design*, 3rd ed., McGraw-Hill Book Co., New York, NY.

7. Example Usage of Natural Computing Graphs

With the scheme for representing graphs in computers described in the foregoing, one can build documents containing a variety of graphs. Such graphs will manifest themselves in three basic forms: (1) graphs embedded in text, (2) graphs that are usable in an interactive mode, and (3) graphs that can be connected into procedures. Figure 24 shows a graph embedded in text; the menus available to manipulate the document as well as the graphs are not shown. When the user wishes to use the graph to obtain values, the graph is changed to an interactive mode, as shown in figure 25, where a highlighted sample graph appears with input and output boxes. When the user types in an input value, result values are output by the system. As stated earlier, this interactive mode is used to test a graph in isolation. An end user uses this mode to obtain values to see the trends or to use the result values in a series of calculations of a temporary nature. Finally, when a domain specialist wishes to incorporate a graph into a procedure, a procedure is invoked, and a graph is connected to that procedure (a procedure is another Natural Computing object, discussed in Karamchetty, 1997, 2000a). Figure 26 shows a procedure consisting of a number of Natural Computing objects. Once a graph is set into a procedure, it is embedded inside the procedure and calculations can be carried out with the entire procedure.

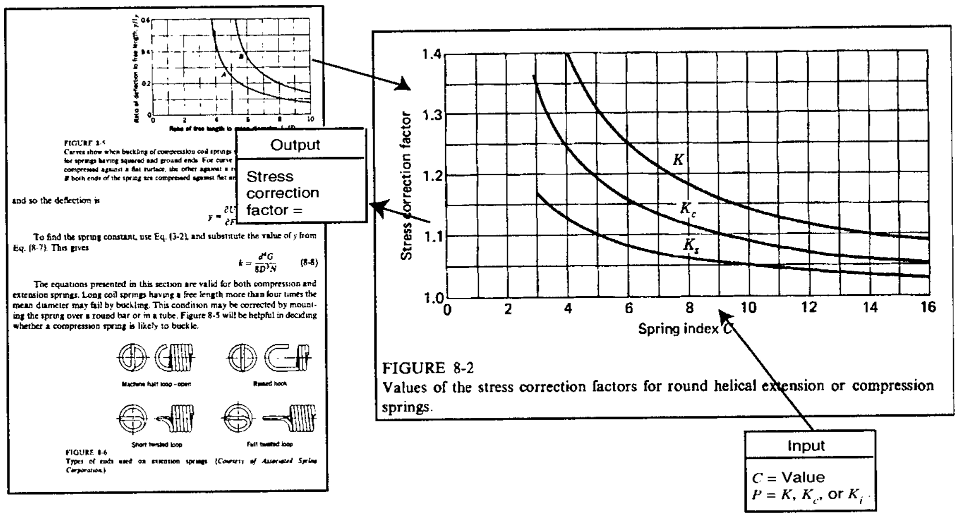


FIGURE 8-5
Curves show when buckling of compression coil springs [illegible] for springs having squared and ground ends. For curve [illegible] compressed against a flat surface, the other against a r[illegible] If both ends of the spring are compressed against flat an[illegible]

and so the deflection is

$$y = \frac{\partial U}{\partial F}$$

To find the spring constant, use Eq. (3-2), and substitute the value of y from Eq. (8-7). This gives

$$k = \frac{d^4 G}{8D^3 N} \qquad (8\text{-}8)$$

The equations presented in this section are valid for both compression and extension springs. Long coil springs having a free length more than four times the mean diameter may fail by buckling. This condition may be corrected by mounting the spring over a round bar or in a tube. Figure 8-5 will be helpful in deciding whether a compression spring is likely to buckle.

FIGURE 8-6
Types of ends used on extension springs. (Courtesy of Associated Spring Corporation.)

FIGURE 8-2
Values of the stress correction factors for round helical extension or compression springs.

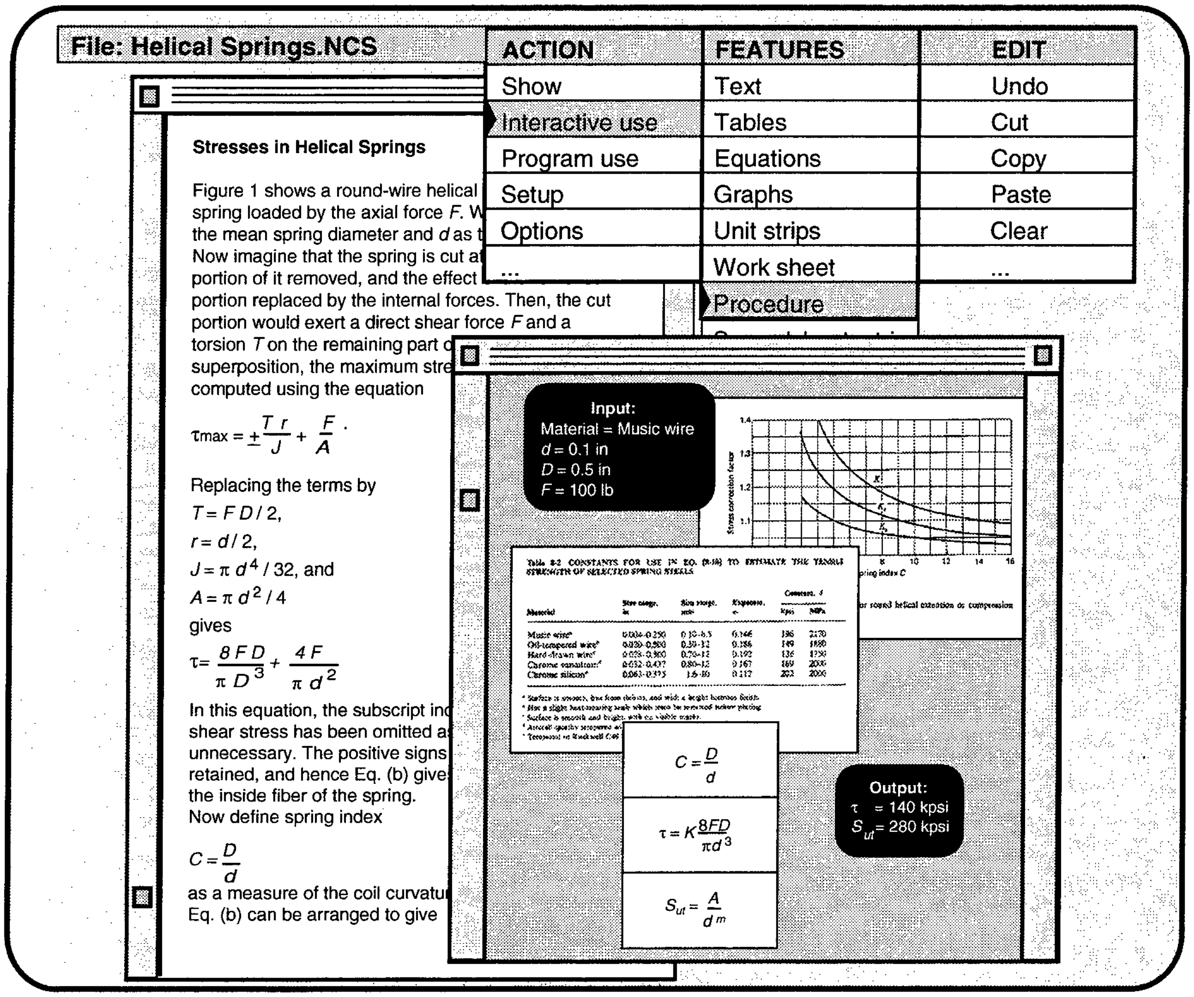
Output

Stress correction factor =

Input

C = Value
$P = K, K_c$, or K_i

Figure 25. Activating a text-embedded graph object for interactive use.



Stresses in Helical Springs

Figure 1 shows a round-wire helical spring loaded by the axial force *F*. W the mean spring diameter and *d* as t Now imagine that the spring is cut at portion of it removed, and the effect portion replaced by the internal forces. Then, the cut portion would exert a direct shear force *F* and a torsion *T* on the remaining part c superposition, the maximum stre computed using the equation

$$\tau_{\max} = \pm\frac{T\,r}{J} + \frac{F}{A}.$$

Replacing the terms by

$T = F\,D\,/\,2,$

$r = d\,/\,2,$

$J = \pi\, d^4\,/\,32,$ and

$A = \pi\, d^2\,/\,4$

gives

$$\tau = \frac{8\,F\,D}{\pi\,D^3} + \frac{4\,F}{\pi\,d^2}$$

In this equation, the subscript inc shear stress has been omitted a unnecessary. The positive signs retained, and hence Eq. (b) give the inside fiber of the spring. Now define spring index

$$C = \frac{D}{d}$$

as a measure of the coil curvatu Eq. (b) can be arranged to give

Input:
Material = Music wire
d = 0.1 in
D = 0.5 in
F = 100 lb

Table 8-2 CONSTANTS FOR USE IN EQ. (8-18) TO ESTIMATE THE TENSILE STRENGTH OF SELECTED SPRING STEELS

Material	Size range, in	Size range, mm	Exponent, m	Constant, A kpsi	Constant, A MPa
Music wire[a]	0.004-0.250	0.10-6.5	0.146	196	2170
Oil-tempered wire[b]	0.020-0.500	0.50-12	0.186	149	1880
Hard-drawn wire[c]	0.028-0.500	0.70-12	0.192	136	1750
Chrome vanadium[d]	0.032-0.437	0.80-12	0.167	169	2000
Chrome silicon[e]	0.063-0.375	1.6-10	0.112	202	2000

[a] Surface is smooth, free from defects, and with a bright lustrous finish.
[b] Has a slight heat-treating scale which must be removed before plating.
[c] Surface is smooth and bright with no visible marks.
[d] Aircraft-quality tempered wi
[e] Tempered to Rockwell C49

$$C = \frac{D}{d}$$

$$\tau = K\frac{8FD}{\pi d^3}$$

$$S_{ut} = \frac{A}{d^m}$$

Output:
τ = 140 kpsi
S_{ut} = 280 kpsi

Figure 26. An example Natural Computing screen showing a procedure consisting of a graph, a table, and equations.

8. Conclusions

Graphs are ubiquitous functional relationships used freely in paper-based documents, such as textbooks, handbooks, journals, newspapers, and flyers. Although graphs are complex representations, people use them relatively effortlessly. I have explained in this report computer representation of graphs for a similar use in a computer calculation system.

The development of a base graph class and several derived classes is the appropriate method to represent and use graphs. I have analyzed and identified the components and structure of a graph. The uniqueness of the graph anatomy presented in this report is that it allows a variety of graphs to be handled. Data can be obtained from graphs by interpolation and extrapolation, where permissible. Display of graphs ensures the presentation to the user of opportunities with given data in a graph and warns the user of pitfalls in the data. The proposed representation and operations allow graphs to behave responsibly and justify their behavior.

Representation and use of graphs in the manner described in this report would mark a giant step for computing, as the data representation follows natural forms. With the availability of such representations and processes, electronic handbooks, textbooks, documents, journals, and flyers can be realized. These electronic documents can be connected seamlessly for performing calculations as if they were paper-based documents. From these various electronic library sources, knowledge can be exchanged and combined, and more powerful, economical, and expeditious computational procedures and systems can be developed.

Acknowledgments

The following companies have given permission to use their copyrighted material, and the author is grateful to them:

- McGraw-Hill Book Co., New York, NY,
- Prentice Hall, Inc., Englewood-Cliffs, NJ,
- Iowa State University Press, Ames, IA, and
- John Wiley and Sons Inc., New York, NY.

References

Baumeister, Theodore, and Lionel S. Marks, eds. (1964). *Mechanical Engineers' Handbook*, 7th ed., McGraw-Hill Book Co., New York, NY.

Coad, Peter, and Edward Yourdon (1991). *Object-Oriented Analysis*, 2nd ed., Yourdon Press Computing Series, Prentice Hall, Englewood Cliff, NJ.

Gartmann, Hans (1970). *The DeLaval Engineering Handbook*, 3rd ed., McGraw-Hill Book Co., New York, NY.

Green, C. R. (1984). *Technical Physics*, Prentice-Hall, Inc., Englewood-Cliffs, NJ.

Karamchetty, Som D. (1997). "Natural Computing," U.S. patent No. 5,680,557, (October 21).

Karamchetty, Som D. (2000a). *Natural Computing: Its Impact on Software Development*, U.S. Army Research Laboratory, ARL-TR-2040.

Karamchetty, Som D. (2000b). *Natural Computing: Analysis of Tables for Computer Representation*, U.S. Army Research Laboratory, ARL-TR-2041.

Karassik, Igor J., William C. Krutzsch, Warren H. Fraser, and Joseph P. Messina (1976). *Pump Handbook*, McGraw-Hill Book Co., New York, NY.

Keenan, Joseph, H., Frederick G. Keyes, Philip G. Hill, and Joan G. Moore (1969). *Steam Tables: Thermodynamic Properties of Water Including Vapor, Liquid, and Solid Phases*, John Wiley and Sons Inc., New York, NY.

Shigley, Joseph E. (1977). *Mechanical Engineering Design*, 3rd ed., McGraw-Hill Book Co., New York, NY.

Van Wylen, Gordon J., and Richard E. Sonntag (1965). *Fundamentals of Classical Thermodynamics*, John Wiley and Sons Inc., New York, NY.

Vaughn, Richard C. (1974). *Quality Control*, Iowa State University Press, Ames, IA.

Wolfram, Stephen (1991). *Mathematica®, A System for Doing Mathematics by Computer*, 2nd ed., Addison-Wesley Publishing Company, Reading, MA.

Distribution

Admnstr
Defns Techl Info Ctr
Attn DTIC-OCP
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

Ofc of the Secy of Defns
Attn ODDRE (R&AT)
The Pentagon
Washington DC 20301-3080

Ofc of the Secy of Defns
Attn OUSD(A&T)/ODDR&E(R) R J Trew
3080 Defense Pentagon
Washington DC 20301-7100

AMCOM MRDEC
Attn AMSMI-RD W C McCorkle
Redstone Arsenal AL 35898-5240

CECOM
Attn PM GPS COL S Young
FT Monmouth NJ 07703

Dir for MANPRINT
Ofc of the Deputy Chief of Staff for Prsnnl
Attn J Hiller
The Pentagon Rm 2C733
Washington DC 20301-0300

TECOM
Attn AMSTE-CL
Aberdeen Proving Ground MD 21005-5057

US Army ARDEC
Attn AMSTA-AR-TD M Fisette
Bldg 1
Picatinny Arsenal NJ 07806-5000

US Army Info Sys Engrg Cmnd
Attn ASQB-OTD F Jenia
FT Huachuca AZ 85613-5300

US Army Natick RDEC
Acting Techl Dir
Attn SSCNC-T P Brandler
Natick MA 01760-5002

US Army Simulation, Train, & Instrmntn Cmnd
Attn J Stahl
12350 Research Parkway
Orlando FL 32826-3726

US Army Soldier & Biol Chem Cmnd
Dir of Rsrch & Techlgy Dirctrt
Attn SMCCR-RS I G Resnick
Aberdeen Proving Ground MD 21010-5423

US Army Tank-Automtv Cmnd Rsrch, Dev, & Engrg Ctr
Attn AMSTA-TR J Chapin
Warren MI 48397-5000

US Army Train & Doctrine Cmnd
Battle Lab Integration & Techl Dirctrt
Attn ATCD-B J A Klevecz
FT Monroe VA 23651-5850

US Military Academy
Mathematical Sci Ctr of Excellence
Attn MDN-A LTC M D Phillips
Dept of Mathematical Sci Thayer Hall
West Point NY 10996-1786

Nav Surface Warfare CtrA
Attn Code B07 J Pennella
17320 Dahlgren Rd Bldg 1470 Rm 1101
Dahlgren VA 22448-5100

DARPA
Attn S Welby
3701 N Fairfax Dr
Arlington VA 22203-1714

Palisades Inst for Rsrch Svc Inc
Attn E Carr
1745 Jefferson Davis Hwy Ste 500
Arlington VA 22202-3402

NASA Langley Rsrch Ctr
Vehicle Techlgy Ctr
Attn AMSRL-VT W Elber
Hampton VA 23681-0001

Distribution (cont'd)

US Army Rsrch Lab
Attn AMSRL-WM I May
Aberdeen Proving Ground MD 21005-5000

US Army Rsrch Lab
Attn AMSRL-RO-D C Chang
Attn AMSRL-RO-EN W Bach
PO Box 12211
Research Triangle Park NC 27709

US Army Rsrch Lab
Attn AMSRL-CI N Radhakrishnan
Aberdeen Proving Ground MD 21005-5067

US Army Rsrch Lab
Attn AMSRL-HR R L Keesee
Aberdeen Proving Ground MD 21005-5425

US Army Rsrch Lab
Attn AMSRL-VP R Bill
21000 Brookpark Rd
Cleveland OH 44135-3191

US Army Rsrch Lab
Attn AMSRL-SL J Wade
White Sands Missile Range NM 88002

US Army Rsrch Lab
Attn AMSRL-DD J Miller
Attn AMSRL-CI-AI-A Mail & Records Mgmt
Attn AMSRL-CI-AP Techl Pub (3 copies)
Attn AMSRL-CI-LL Techl Lib (3 copies)
Attn AMSRL-IS J D Gantt
Attn AMSRL-IS-CB L Tokarcik
Attn AMSRL-IS-CD P Jones
Attn AMSRL-IS-CI B Broome
Attn AMSRL-IS-CS G Racine
Attn AMSRL-IS-D COL M R Kindl
Attn AMSRL-IS-D P Emmerman
Attn AMSRL-IS-D R Slife
Attn AMSRL-IS-E D Brown
Attn AMSRL-IS-TA J Gowens
Attn AMSRL-SE J Pellegrino
Attn AMSRL-SE-EP S Karamchetty
(30 copies)
Attn AMSRL-ST C I Chang
Adelphi MD 20783-1197

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE February 2000	3. REPORT TYPE AND DATES COVERED Final, 1995–1997

4. TITLE AND SUBTITLE Natural Computing: Analysis of Graphs for Computer Representation	5. FUNDING NUMBERS DA PR: N/A PE: N/A
6. AUTHOR(S) Som Karamchetty	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Attn: AMSRL-IS-C email: skaramch@arl.mil 2800 Powder Mill Road Adelphi, MD 20783-1197	8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-2042
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory 2800 Powder Mill Road Adelphi, MD 20783-1197	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
ARL PR: N/A
AMS code: N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Some fundamental objects in practical documents have not been implemented in software so that they can be used easily for calculation. One such object is the graph. A survey of practical graphs found in a variety of real-world documents reveals that many of their useful features are not captured in software. I describe the salient characteristics and features of graphs and propose data structures and computer representations for graph objects. Through the adoption of such structures and representations, practical graph objects could be developed for use by domain specialists. Such graphs embedded in electronic documents can be used in interactive applications to retrieve data, but most importantly, they could be used as functional representations for "cutting and pasting" in procedures and programs. Use of these graph objects, together with other natural computing objects (such as equations, tables, and procedures), will permit electronic documents like handbooks, textbooks, journals, and bulletins to be used seamlessly for calculations by both domain specialists and naive users. Such developments will reduce the lag between information availability and its use in calculations, encouraging the further development of knowledge. The face of software development for computation will change, and many of the software engineering costs will be contained.

14. SUBJECT TERMS Natural computing, software engineering, object-oriented programming	15. NUMBER OF PAGES 43
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102